https://github.com/kharmajbird/quest.git

(for Dockerfile & node_modules friends)

https://github.com/kharmajbird/quest-docs

(what you are reading now, along with screen shots)

The sheer size of the quest source repository caused numerous issues that I had to work around.

I first stood up the app in my local Docker environment, with all of the dependencies (express, etc) in place.

The app basically refused to run, stating that it was not running in a cloud environment. Screenshots attached.

Well, at least that portion worked.

The problem is this node app is distributed as compiled code; something that I had never seen before. Doing a 'file' command on the binaries show that the 00*js files are actually Linux binaries. They run just fine under Docker, but there is no way to view the code, or modify it.

My initial attempt to deploy the code was via ECR. The upload of my docker image to the registry did not quite work, for various reasons. Perhaps the file size. Permissions are fine. Unknown, moving on to the next technology choice.

My next effort was to use Terraform, in order to lay out the infrastructure in detail to be highly-available. I used a template that has served me well in the past, but for this particular one, I would have to create a custom AMI image with Packer, and troubleshoot all of that down the rabbit hole. Too much time.

The TF route still would have been fun to follow through with; maybe another time….

Elastic Beanstalk is probably the best method to solve this sort of challenge.

It takes care of all of the load-balancing, multiple region support, public-facing URL and other things; all one has to do is upload a *.zip of your application, choose the platform (node.js) and launch it.

In theory.

I tried multiple times to deploy this compiled node.js app into Elastic Beanstalk, but no matter how I did it, I kept coming up with a 502 error when trying to access the endpoint, and the health checks were coming back as severe.

My deployment method for this code was sound, as I was able to deploy a hello world node.js app into EBS, using the same zip format as the downloaded quest zip, and access it without a problem.

It just seems to be a difference between the obfuscated compiled quest code and human-readable code and yaml and such.

Oh, and please see the Dockerfile within the kharmajbird/quest repo I provided; it shows how to insert the secret word as an environment variable to a running docker container.

That worked quickly, but once the compiled code found it wasn’t running in a cloud environment, it simply bailed, with no opportunity to tweak the code to get around that constraint, since it was compiled and not scripted.

I thought about running an strace on the container, in my local docker environment, but no… too little ROI :(

Elastic Beanstalk should be the way to go to deploy this quest app into AWS.

But having not used it before, I think I’m missing a security group setting or something that prevents me from getting past that 502 error.

I would need another set of eyes :)